International Journal of Computer Networking,
Wireless and Mobile Communications (IJCNWMC)
ISSN (P): 2250–1568; ISSN (E): 2278–9448
Vol. 11, Issue 2, Dec 2021, 1 – 8

IDENTIFICATION AND RECOVERING LOCALITY EXPLOITATIONS IN DISTRIBUTED SDN CONTROLLERS

V. R. SUDARSANA RAJU & DR. RUBINI P

CMRU, Bangalore, India

ABSTRACT

Software Defined Networking is a new approach, which will have many advantages comparing with legacy networking. This SDN will address many issues and bring in more capabilities into the field of networking. Separation of Control plane and data plane is one of the important features of the architecture, which will bring in many applications, can be run in a centralized location called the controller. Centralized controller can achieve load balancing with help of data plane under those controllers. It minimized the effects of high traffic and performance related issues, resolving the issues in the network according to the technique used in each algorithm. The main problem addressed to provide high performance to SDN networks is efficient distribution of network resources, by considering the characteristics, such as throughput and the performance of each device, which will resolve the issues during network operations, high network utilization and server utilization, etc.

KEYWORDS: Distributed Software Defined Networking Controllers

Received: Oct 20, 2021; **Accepted**: Nov 10, 2021; **Published**: Nov 25, 2021; **Paper Id.**: IJCNWMCDEC20211

Original Article

1. INTRODUCTION

Software Defined Networking (SDN)is the concept of representing a physical network in a virtual manner. It will have many advantages in terms of operation and maintenance. Separation of control plane and data plane, combing decision-making logic and taking control of networking can be done in central location called controller. In this controller, on need basis, different modules and applications can be run to regulate packet flow. Multiple such types of controllers are connected, issues raised in such types of networks can be recovered and identified with the help of algorithms presented in this paper. Using these algorithms, any link failure issues and performance issues can be resolved without any manual interventions. Separation of control plane and data plane is the main feature of the SDN. The communication method used to communicate between the data and control plane is Open Flow, this is the standard, which will be used across all controllers. Well defined North bound and South bound APIs are available for interaction and communication within these controllers. Proposed algorithms will help to resolve any issues and performance issues across these controllers. These controllers are will be designed using monolithic networking operating systems, which will depend on languages like C, C++, Perl, Python, Java, etc. Identifying the failure and recovering the controllers with link failures, etc will be handled by the proposed algorithms.

Need to identify the failures of any specific controller which may lead to bringing down performance, such issues need to be addressed and there should be some automatic algorithmic recovery mechanism to a possible extent without manual intervention, to address this kind of failures in distributed SDN network by exploiting location of the failure.

2. FUNCTIONS OF SDN CONTROLLERS

Separation of control and data plane is the basic feature in SDN, entire intelligence of the network, which will be part of the control plane, moved to the controller; all decision making will happen, many features and applications can be added to these controllers based on the requirement. Open Flow is a communication standard between the control plane and data plane for communication to address flow table issues to redirect the packets in the data plane. Link failure collection is function of one of the modules in controller and failures in these links will be communicated to controllers for recovery using the algorithms proposed.

3. DYNAMIC LOAD BALANCING ALGORITHMS ON SDN NETWORKS

SDN networks can be applicable to the control and data planes, to minimize the issues raised in the networks, according to the technique used in each algorithm. The main problem addressed in the data plan in order to provide high performance to SDN networks is efficient resource distribution, taking into account their characteristics, such as throughput and the performance of each device. The following algorithms proposed are used to identify and recover any controller failures in distributed SDN controllers.

Aalgorithm-1 Calculate weight_Delay

- 1:functionCALUCULATEWEIGHT_DELAY(Path)
- 2: if Weight.updated() then
- 3: ReadTable(Path)
- 4: else
- 5: for Node do 0 size(CurrentPath)
- 6: tx ←--- transmitted time
- 7: rx←--- received time
- 8: DelayRate ←---- rx - tx
- 9: timesleep(1)
- 10: C ←---- capacitySwitch()
- 11: U ←---- DelayRate
- 12: WD ←--- u/c*100
- 13: end for
- 14: Tableupdate(Path,wd)
- 17: end if
- 18: return wd
- 19: end function

Algorithm-2 Check for weight_delay updated or not

- 1: function CHECK_DELAY(Path)
- 2: Row←--Path.split(":",1)[0]
- 3: Column←---Path.split(":",1)[1]
- 4: TableTime←--Table[row,column,1]
- 5: CurrentTime←--DateTime.now()
- 6: if (CurrentTime - TableTime)<=10s then
- 7: status ←-- True
- 8: else:
- 9: status←-- False
- 10: endif
- 11: return sttus
- 12: end function

Algorithm- 3 Table Update for Delay

- 1: Procedure TABLEUPDATEDELAY (path, weight_delay)
- 2: Row←--Path.split()":",1[0]
- 3: Column ←--path.split(":",1)[1]
- 4: Currentime ←--Datetime.now()
- 5: Table[row, column,1]←--Currenttime
- 6: Table[row,column,1]←--Currenttime
- 7: end Procedure

These algorithms will be used to calculate the difference in delay and update data in table for recovery.

Algorithm- 4 Calculate weight_throughput

- 1:function CALUCULATEWEIGHT_Throughput(Path)
- 2: if Weight.updated() then
- 3: ReadTable(Path)
- 4: else
- 5: for Node do 0 size(CurrentPath)

- 6: tx ←--- transmitted time
- 7: rx ←--- received time
- 8: ThroughputDiff←---- rx - tx
- 9: timesleep(1)
- 10: C ←---- capacitySwitch()
- 11: U ←---- ThroughputDiff
- 12 WT←--- u/c*100
- 13: end for
- 14: Tableupdate(Path,wt)
- 17: end if
- 18: return wt
- 19: end function

Algorithm-5 Check for weight_throughput updated or not

- 1: function CHECK_THROUGHPUT(Path)
- 2: Row←--Path.split(":",1)[0]
- 3: Column←---Path.split(":",1)[1]
- 4: TableTime←--Table[row,column,1]
- 5: CurrentTime←--DateTime.now()
- 6: if (CurrentTime - TableTime) <=10s then
- 7: status ←-- True
- 8: else:
- 9: status←-- False
- 10: endif
- 11: return sttus
- 12: end function

Algorithm- 6 Table Update for Throughput

- 1:Procedure TABLEUPDATETHROUGHPUT (path, weight_delay)
- 2: Row←--Path.split()":",1[0]

- 3: Column ←--path.split(“:”,1)[1]
- 4: Currentime ←--Datetime.now()
- 5: Table[row, column,1]←--Currenttime
- 6: Table[row,column,1]←--Currenttime
- 7: end Procedure

These algorithms are used to calculate the difference in latency and update data in table for recovery.

Algorithm- 7 Calculate weight_Latency

- 1:function CALUCULATE WEIGHT_Latency(Path)
- 2: if Weight.updated() then
- 3: ReadTable(Path)
- 4: else
- 5: for Node do 0 size(CurrentPath)
- 6: tx ←--- transmitted time
- 7: rx ←--- received time
- 8: LatencyRate←---- rx - tx
- 9: timesleep(1)
- 10: C ←---- capacitySwitch()
- 11: U ←---- LatencyRate
- 12: WD ←--- u/c*100
- 13: end for
- 14: Tableupdate(Path,wl)
- 17: end if
- 18: return wl
- 19: end function

Algorithm- 8Check for weight_latency updated or not

- 1: function CHECK_LATENCY(Path)
- 2: row←--Path.split(“:”,1)[0]
- 3: column←---Path.split(“:”,1)[1]

- 4: TableTime←--Table[row,column,1]
- 5: CurrentTime←--DateTime.now()
- 6: if (CurrentTime - TableTime) <=10s then
- 7: status ←-- True
- 8: else:
- 9: status←-- False
- 10: endif
- 11: return sttus
- 12: end function

Algorithm- 9 Table Update for Latency

- 1: Procedure TABLEUPDATELATENCY (path, weight_delay)
- 2: row←--Path.split()":",1[0]
- 3: column ←--path.split(":",1)[1]
- 4: Currentime ←--Datetime.now()
- 5: Table[row, column,1]←--Currenttime
- 6: Table[row,column,1]←--Currenttime
- 7: end Procedure

These algorithms are used to calculate the difference in latency and update data in table for recovery.

Weight_delay

Weight_throughput

Weight_latency

W= Weight_delay+Weight_throughput+eight_latency

N – Size of network. N x N data matrix will be created with corresponding attribute and time.

Entire data will be stored in XML format and every 5 minutes this data will be over written.

Statistics will be calculated and monitored by the algorithm. If it exceeds the threshold limit, the load balancer module will be triggered.

Load balancer algorithms consider the overload and capacity of each device within the network. The proposed model aims to improve the performance of SDN networks and ensure quality in a dynamic network environment.

4. CONCLUSIONS

Software Defined Networks are used for a variety of purposes and soon legacy networks will be slowly migrated to this new technology where they will work together during this transition.

This work presented the application of a load balancer algorithm in SDN through simulations performed with Mininet. The proposed algorithms will be used to recover the network issues in distributed controller environment so that network performance can be increased and enhanced.

REFERENCES

Books:

1. *Thomas D. Nadeau & Ken Gray (2016), SDN: Software Defined Networks, O`Reilly, Beijing.*

Websites

2. *https://onosproject.org/*
3. *D. Kreutz F. M. V. Ramos P. Esteves C. Esteve S. Azodolmolky "Software-Defined Networking: A Comprehensive Survey" Proceedings of the IEEEvol. 103 pp. 10-13 2016.*
4. *Q. Y. Zuo M. Chen G. S. Zhao C. Y. Xing G. M. Zhang P. C. Jiang "Research on OpenFlow-based SDN technologies" Journal of Software vol. 24 pp. 1078-1097 2015.*
5. *S. Jain A. Kumar et al. "B4: experience with a globally-deployed software defined wan" ACM SIGCOMM 2016 Conference on SIGCOMM. pp. 3-14 2016.*
6. *A. Singh J. Ong A. Agarwal et al. "Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network" ACM Conference on Special Interest Group on Data Communication pp. 183-197 2015.*
7. *C. K. Zhang Y. Cui H. Y. Tang J. P. Wu "State-of-the-Art survey on software-defined networking (SDN)" Journal of Software vol. 26 pp. 62-81 2015.*
8. *T. Q. Zhou Z. P. Cai J. Xia M. Xu "Traffic engineering for software defined networks" Journal of Software vol. 27 pp. 394-417 2016.*
9. *S. Scott-Hayward S. Natarajan S. Sezer "A Survey of Security in Software Defined Networks" IEEE Communications Surveys & Tutorials vol. 18 pp. 623-654 2016.*
10. *A. Blen A. Basta M. Reisslein "Survey on Network Virtualization Hypervisors for Software Defined Networking" IEEE Communications Surveys & Tutorials vol. 18 pp. 655-685 2016.*
11. *Rajeev, K., and K. Supraja. "A Study of Intelligent Controllers Application in Distributed Systems."International Journal of Mechanical Engineering (IJME)5.4, Jun - Jul 2016; 53-62*
12. *Sherasiya, Tariqahmad, Hardik Upadhyay, and Hiren B. Patel. "A survey: Intrusion detection system for internet of things." International Journal of Computer Science and Engineering (IJCSE) 5.2 (2016): 91-98.*
13. *Gnanaraj, F. Fredrick, and KR Vijaya Kumar. "Analysis of Vibration Detection using Active Controller in the Smart Cantilever Composite Beam with LQR and Fuzzy Techniques." International Journal of Mechanical and Production Engineering Research and Development (IJMPERD) 8. 5, Oct 2018, 67 76.*

14. *Yedilkhan, Amirgaliyev, et al. "Predicting heating time, thermal pump efficiency and solar heat supply system operation unloading using artificial neural networks." International Journal of Mechanical and Production Engineering Research and Development 9.6 (2019): 221-232.*